



3 1176 00161 6888

NASA CR-165, 730

NASA Contractor Report 165730

STAR ADAPTATION OF QR ALGORITHM

NASA-CR-165730
1981 00180 41

Shantilal N. Shah

HAMPTON INSTITUTE
Hampton, Virginia 23668

LIBRARY COPY

June 1981

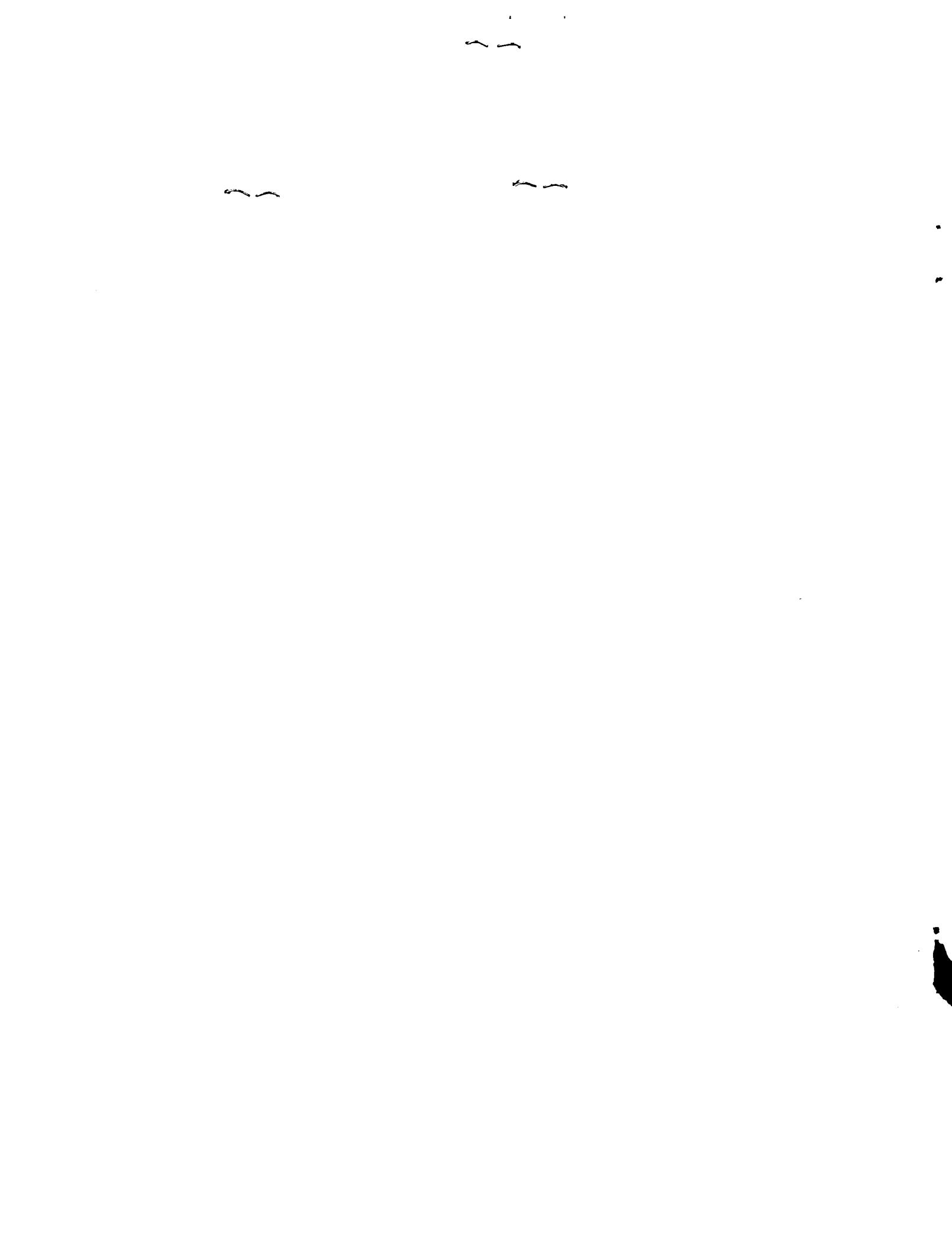
JUN 10 1981

LANGLEY RESEARCH CENTER
LIBRARY, NASA
HAMPTON, VIRGINIA



National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665



SUMMARY

The QR algorithm used to solve over-determined systems of linear equations was adapted to execute efficiently on the Control Data STAR-100 computer. Using the new vectorized algorithm, the STAR-100 computer solved a system of 250 equations in 50 unknowns in less than 8.5% of the time it took using the original scalar version. This paper describes how the scalar program was adapted for the STAR-100 and indicates why these adaptations yielded an efficient STAR program. Program listings of the old scalar version and of the new vectorized SL/1 version are presented in the appendices. Execution times for the two versions, applied to the same system of linear equations, are compared.

INTRODUCTION

Programs written in standard FORTRAN language for serial computers do run on the Control Data STAR-100 computer, but very inefficiently. To take advantage of the architecture and vector-processing capabilities of the STAR-100 computer it is necessary to vectorize the algorithms in these programs. Frequently one must rearrange the data and computations. This paper describes how the QR algorithm to solve over-determined systems of linear equations was vectorized and what factors were considered in developing an efficient STAR program.

The vectorized program utilizes SL/1, a high level language developed by NASA's Langley Research Center for the STAR-100 computer. SL/1 incorporates many features designed to see that programs it compiles take full advantage of the STAR's architecture and capabilities, including half-word storage and arithmetic. SL/1 is compatible with FORTRAN in the sense that programs written in either language can call subroutines written in the other. In utilizing the program presented in this paper, familiarity with some of the notations used in the SL/1 language will be helpful.

N81-26779 #

General suggestions concerning the adaptations of algorithms for efficient use on the STAR may be found in the paper, "Star Adaptation for Two Algorithms used on Serial Computer," by Lona M. Howser and Jules J. Lambiotte (see ref. 1).

ADAPTATION OF QR ALGORITHM TO SOLVE OVER-DETERMINED SYSTEMS OF LINEAR EQUATIONS

The NASA computer mathematics library presently has a subroutine called QRASOS, written in FORTRAN for serial computers, to solve an over-determined system of linear equations. This subroutine decomposes the matrix A of the system $AX=B$ using Householder transformations. (For details of this algorithm see ref. 2). To compute these transformations, it uses subroutines SAXPY, SSCAL, SCOPY and function SDOT from the Basic Linear Algebra Subroutines (BLAS). For an $m \times n$ ($m \geq n$) matrix A it makes n^2 calls to these subroutines and functions to solve the given system. Subroutine calls are very expensive on the STAR-100 computer.

In SL/1, a matrix can be stored either column-wise or row-wise. Column storage means that elements in one column of the matrix are stored as one vector (contiguous locations). Similarly, row-storage means that elements in one row are stored as one vector (contiguous locations). In vectorizing this algorithm both row and column storage of matrices A and B were tried.

With row-wise storage, reordering of the scalar-version computations is required but use of the inner product macro to decompose matrix A is avoided. With column-wise storage the computational steps are the same as in the scalar versions with vector instructions replacing scalar instructions, but use of the dot product macro is required. It was expected that, because of the avoidance of the dot product macro, the row-wise approach would offer a considerable saving in CPU time.

Test results show that in using the STAR computer for this algorithm both

vectorized versions offer considerable CPU time savings over the scalar program, but that contrary to expectations column-wise storage is more efficient than row-wise storage (see table).

Size of Matrix A	CPU TIME IN SECONDS TO DECOMPOSE		CPU TIME IN SECONDS TO DECOMPOSE AND SOLVE	
	New Vectorized Version (Column Storage)	New Vectorized Version (Row Storage)	Old Scalar Version	New Vectorized Version (Column Storage)
250 x 200	2.169	2.695	26.34	2.239
120 x 100	.405	.588	3.396	.433
250 x 50	.158	.863	2.223	.178
100 x 10	.006	.069	0.056	.009
200 x 30	.053	.416	0.698	.063

Algorithms for both row-wise storage and column storage to decompose the matrix A are given. Back substitutions are not discussed here. In both algorithms, A is an $m \times n$ matrix and WK is a vector of length n.

COLUMN STORAGE

When matrix A is stored column-wise, the decomposition of A is achieved as follows: (Note: All references to i^{th} column refer to column entries on and below the diagonal).

- (1) Take the inner product of i^{th} column with itself and store in WK_i
- (2) Take the square root of WK_i
- (3) If $\text{WK}_i = 0$, go to step (10)
- (4) $\text{WK}_i = \text{WK}_i \times \text{Sign of } A_{i,i}$
- (5) Divide column i by WK_i
- (6) Add 1 to $A_{1,i}$

These 6 steps compute the Householder transformation for column i.

To apply this transformation to the columns $K = i+1, \dots, n$ do the following steps:

- (7) Take the inner product of column i with column K and store the result in t
- (8) Divide t by $A_{i,i}$ and then store the negative of the result in t
- (9) Multiply column i by t and add the result to column K
- (10) Store $A_{i,i}$ in t, $-W_{K_i}$ in $A_{i,i}$ and t in W_{K_i}

When $i=n$ perform steps 1 thru 6 and 10.

ROW STORAGE

When the matrix A is stored row-wise, the decomposition of A is achieved as follows: (Note: In the steps 1 thru 6 below, all references to the row j in the i^{th} step of decomposition refer to the entries on and to the right of the diagonal. All references to the vector WK refer to its $i^{th}, (i+1)^{th}, \dots, n^{th}$ elements. In steps 7 thru 10 all references to the row j in the i^{th} step of decomposition refer to the entries to the right of the diagonal and all references to the vector WK refer to its $(i+1)^{th}, (i+2)^{th}, \dots, n^{th}$ elements).

At i^{th} step of decomposition ($i=1, 2, \dots, n-1$).

- (1) Set $W_K=0$
- (2) For $j=1, 2, \dots, m$, multiply row j by $A_{j,i}$ and add the result to W_K
- (3) Take the square root of W_{K_i}
- (4) If $W_{K_i}=0$ go to step 11
- (5) Multiply W_{K_i} by sign of $A_{i,i}$
- (6) Divide $A_{i,i}$ by W_{K_i} and add 1 to the result
- (7) Divide W_K by W_{K_i} and add row i of A to W_K
- (8) Divide W_K by $-A_{i,i}$
- (9) For $j=i+1, i+2, \dots, m$
Divide $A_{j,i}$ by W_{K_i}

(10) For $j=i, i+1, \dots, m$, multiply WK by $A_{j,i}$ and add the result to row j of A

(11) Store $A_{i,i}$ in t , $-WK$ in $A_{i,i}$ and t in WK_i

When $i=n$, perform Steps 1 thru 6 and 11.

WHY ROW-STORAGE IS SLOWER THAN COLUMN-STORAGE

As pointed out earlier, if the matrix A is stored row-wise, the use of the inner product macro is avoided and the computation of the Householder transformations and their application to other columns at each step of the decomposition is accomplished by the use of a vector multiplication by a scalar and then a vector addition. This should result in a considerable savings of the CPU time for a large matrix. However, our numerical experiments show just the opposite. This can be explained as follows: When an $m \times n$ ($m \geq n$) matrix A is stored row-wise, the vector lengths in that algorithm are proportional to n , the smaller dimension. On the other hand, for column-wise storage the vector lengths are proportional to m , the larger dimension. Equivalently, we see that the row-stored algorithm requires more vector start-ups ($((m-n)(m-n+1))/2$ more) to do the same number of total computations as the column-stored algorithm, thus requiring more CPU time to do the same amount of work.

Another factor which makes the row-stored algorithm slower is that the transformation elements are stored in the columns of the decomposed matrix. If the matrix is stored row-wise, this leads to additional scalar computations, notably in step 9 of the algorithm. This slows down the computations considerably. Also, if m is large, then not all m vectors in row-wise storage reside in the memory at the same time. Because of need to reference different columns at different steps of algorithm, this could lead to excessive paging. Thus, any advantage gained by avoiding the use of the inner product in the row-wise storage is offset by the need to perform many scalar operations, more iterations and excessive paging.

REFERENCES

Lona M. Howser and Jules J. Lambiotte, Jr., "STAR Adaptation for Two Algorithms Used on Serial Computer," NASA TM X-3003. 1974

J. H. Wilkinson and Reinsch, Linear Algebra, Springer-Verlag, Berlin, 1971 .

APPENDIX A

SL/1 Coding of QR Algorithm

```

MODULE M1      ;      $OPT=1,$SOURCE(1.72);
***** */ */
/* */ */
/*PURPOSE */ */
/* TO SOLVE M SIMULTANEOUS EQUATIONS IN N UNKNOWNNS WITH IP */ */
/* RIGHT HAND SIDES SO THAT THE SOLUTIONS ARE THE BEST POSSIBLE*/ */
/* FIT IN THE LEAST SQUARES SENSE. THE ROUTINE USES HOUSE- */ */
/* HOLDER TRANSFORMATIONS TO PERFORM THE QR DECOMPOSITION */ */
/* OF THE COEFFICIENT MATRIX. */ */
/* */ */
/*USE */ */
/* */ */
/* CALL Q4QRASOS(MAXM,MAXN,M,N,IP,A,B,WT,JOB,X,RSD,SUM,WK,IERR) */ */
/* */ */
/*PARAMETERS */ */
/* */ */
/* MAXM AN INPUT INTEGER SPECIFYING THE FIRST DIMENSION OF THE */ */
/* A,B, AND RSD ARRAYS IN THE CALLING PROGRAM. MAXM MUST */ */
/* BE GREATER THAN OR EQUAL TO M. */ */
/* */ */
/* MAXN AN INPUT INTEGER SPECIFYING THE FIRST DIMENSION OF THE */ */
/* X ARRAY IN THE CALLING PROGRAM. MAXN MUST BE GREATER */ */
/* THAN OR EQUAL TO N. */ */
/* */ */
/* M AN INPUT INTEGER SPECIFYING THE NUMBER OF ROWS OF THE */ */
/* A AND B ARRAYS. M MUST BE GREATER THAN OR EQUAL TO N. */ */
/* */ */
/* N AN INPUT INTEGER SPECIFYING THE NUMBER OF COLUMNS OF */ */
/* THE A ARRAY. */ */
/* */ */
/* IP AN INPUT INTEGER SPECIFYING THE NUMBER OF COLUMNS OF */ */
/* THE B ARRAY. */ */
/* */ */
/* A AN INPUT/OUTPUT TWO-DIMENSIONAL ARRAY WITH FIRST DIMEN- */ */
/* SION EQUAL TO MAXM AND SECOND DIMENSION AT LEAST N. */ */
/* ON INPUT, A MUST CONTAIN THE MATRIX OF COEFFICIENTS OF */ */
/* THE SYSTEM OF EQUATIONS. ON OUTPUT, A CONTAINS INFOR- */ */
/* MATION DESCRIBING THE QR DECOMPOSITION OF A. */ */
/* */ */
/* B AN INPUT TWO-DIMENSIONAL ARRAY WITH FIRST DIMENSION */ */
/* EQUAL TO MAXM AND SECOND DIMENSION AT LEAST IP. */ */
/* THE COLUMNS OF B MUST CONTAIN THE IP RIGHT HAND SIDE */ */
/* VECTORS. */ */
/* */ */
/* WT AN INPUT ONE-DIMENSIONAL ARRAY OF WEIGHTS. IT MUST */ */
/* HAVE LENGTH AT LEAST M. IF WEIGHTING IS DESIRED, */ */
/* THE FIRST M LOCATIONS MUST CONTAIN REAL NUMBERS GREATER */ */
/* THAN ZERO. IF WEIGHTING IS NOT DESIRED, WT [1] MUST BE */ */
/* A NEGATIVE REAL NUMBER. */ */
/* */ */
/* JOB AN INPUT INTEGER SPECIFYING RESULTS TO BE COMPUTED. */ */
/* */ */
/* -1 COMPUTE SOLUTIONS ONLY. */ */
/* -2 COMPUTE RESIDUALS ONLY. */ */
/* -3 COMPUTE BOTH SOLUTIONS AND RESIDUALS. */ */
/* */ */
/* X AN OUTPUT TWO-DIMENSIONAL ARRAY CONTAINING THE SOLU- */ */
/* TIONS. X MUST BE DIMENSIONED WITH FIRST DIMENSION */ */
/* EQUAL TO MAXN AND SECOND DIMENSION AT LEAST IP. IF */ */
/* SOLUTIONS ARE DESIRED INTO MATRIX B THEN MAXN MUST BE */ */
/* EQUAL TO MAXM FOR THIS PARTICULAR CASE. */ */

```

```

/*
 * RSD AN OUTPUT TWO-DIMENSIONAL ARRAY CONTAINING THE RESID- */
 * UALS. RSD MUST BE DIMENSIONED WITH FIRST DIMENSION */
 * EQUAL TO MAXM AND SECOND DIMENSION AT LEAST IP. */
 */

/*
 * SUM AN OUTPUT ONE-DIMENSIONAL ARRAY CONTAINING THE WEIGHTED */
 * SUMS OF SQUARES OF THE RESIDUALS. SUM MUST BE DIMEN- */
 * SIONED AT LEAST IP. */
 */

/*
 * WK A ONE-DIMENSIONAL WORK ARRAY WHICH MUST BE DIMENSIONED */
 * AT LEAST N. ON OUTPUT, WK CONTAINS INFORMATION ON THE */
 * QR DECOMPOSITION OF A. */
 */

/*
 * IERR AN INTEGER ERROR CODE. */
 */

/*
 * -0 NO ERROR DETECTED. */
 * -1 N IS GREATER THAN M. */
 * -2 THE DECOMPOSED MATRIX IS SINGULAR. */
 * -3 WEIGHTING WAS REQUESTED AND ONE OR MORE WEIGHTS */
 * IS NEGATIVE. */
 */

/*
 * SOURCE HAMPTON INSTITUTE, HAMPTON VA. */
 */

/*
 * LANGUAGE SL/I. */
 */

/*
 * DATE RELEASED JANUARY 18, 1980. */
 */

```

```

***** ENTRY PROCEDURE Q4GRASOS (MAXM,MAXN,M,N,IP,A,B,NT,JOB,X,RSD,
      SUM,WK,IERR);
      REAL VECTOR [MAXM] ARRAY(N) A;
      REAL VECTOR [MAXN] ARRAY(IP) X;
      REAL VECTOR [MAXM] ARRAY(IP) B,RSD;
      REAL VECTOR [M] WT;
      REAL VECTOR [N] WK;
      REAL VECTOR [IP] SUM;
      AUTOMATIC REAL T;
      INTEGER I,J,K,L,M,N,IP,MAXM,IERR,MAXN,JOB;
      /*

      /* CHECK FOR M LESS THAN N. */

      /* IF M < N THEN IERR:=-1;
         GO TO LAB1 */

      /* ELSE */

      /* CHECK FOR WEIGHTING */

      /* IF WT[1] >=0 THEN */

      /* CHECK FOR ILLEGAL WEIGHTS */

      /* I:= SELLT(WT,0);
         IF I < M THEN IERR:=-3;
         GO TO LAB1 */

      /* ELSE
         NT[1,M]:= SORT(WT[1:M]);
         FOR I:=1 TO N DO
            AC(I)[1:M]:=A(I)[1:M]*WT[1:M];
         ENDF;
         FOR I:=1 TO IP DO
            BC(I)[1:M]:=B(I)[1:M]*WT[1:M];
         ENDF. */

```

```

        ENDI;
ENDI;
ENDI;

/*
/* CALL Q4SQRDC TO DECOMPOSE MATRIX A.                                */
/*
CALL Q4SQRDC(A,MAXM,M,N,WK);                                         */
/*
/* CALL Q4SQRSL TO SOLVE IP RIGHT HAND SIDES.                         */
/*
CALL Q4SQRSL(MAXM,MAXN,M,N,IP,A,B,WT,JOB,X,RSD,SUM,WK,IERR);      */
IF IERR>0 THEN IERR:=2
ENDI;
LAB1: ENDP;
******/                                                               */
/*
PROCEDURE Q4SQRDC (A,MAXM,M,N,WK);
    REAL VECTOR [MAXM] ARRAY(0) A;
    REAL VECTOR [N] WK;
    AUTOMATIC REAL T;
    INTEGER I,J,K,L,M,N,IP,MAXM,IERR,MAXN,JOB,
/*
/* COMPUTE HH TRANSFORMATION FOR COLUMN I                           */
/*
FOR I:=1 TO N-1 DO
    WK[I]:= ACID[I:MJ].DOT. ACID[I:MJ];
    WK[I]:= SQRT(WK[I]);
    IF WK[I] > 0 THEN
        WK[I]:= WK[I]*ABS(ACID[I])/ACID[I];
        ACID[I:MJ]:=ACID[I:MJ]/WK[I];
        ACID[I]:=ACID[I]+1.;

/*
/* APPLY HH TRANSFORMATION TO REST OF THE COLUMNS                   */
/*
        J:= I+1;
        FOR K:= J TO N DO
            T:= ACID[I:MJ].DOT. ACKD[I:N];
            T:= -T/ACID[I];
            ACKD[I:N]:= ACKD[I:N] + T* ACID[I:N];
        ENDF;
    ENDI;
ENDP;
/*
WK[N]:= ACHD[N:MJ].DOT. ACHD[N:MJ];
WK[N]:= SQRT(WK[N]);
IF WK[N] > 0 THEN
    WK[N]:= WK[N]*ABS(ACHD[N])/ACHD[N];
    ACHD[N:MJ]:= ACHD[N:MJ]/WK[N];
    ACHD[N]:= ACHD[N]+1;
ENDI;
ENDP;
*/
/*
PROCEDURE Q4SQRSL (MAXM,MAXN,M,N,IP,A,B,WT,JOB,X,RSD,
    SUM,WK,IERR);
    REAL VECTOR [MAXM] ARRAY(0) A;
    REAL VECTOR [MAXN] ARRAY (IPI) X;
    REAL VECTOR [MAXM] ARRAY (IPI) B,RSD;
    REAL VECTOR [N] WK;
    REAL VECTOR [IPI] SUM;
    INTEGER I,J,K,L,M,N,IP,MAXM,IERR,MAXN,JOB;
    AUTOMATIC REAL T;
    IERR:=0;

```

```
/*
 * SPECIAL ACTION WHEN M=1
 */
/*
```

11

```
IF M = 1 THEN
    IF WK[1] = 0 THEN
        IERR:=1;
        GO TO LAB4
    ENDI;
    IF JOB <>2 THEN
        FOR I:=1 TO IP DO
            X(I)[1]:= B(I)[1]/A(1)[1];
        ENDF;
    ENDI;
    IF JOB <> 1 THEN
        RSD(1)[1:IP]:=0.0;
    ENDI;
    GO TO LAB4;
ENDI;
```

```
/*
 * COMPUTE TRANS(Q)*B
 */
/*
```

```
FOR I :=1 TO N DO
    IF WK[I] <> 0 THEN
        FOR J:=1 TO IP DO
            T:= A(I)[I:M].DOT. B(J)[I:M];
            T:=-T/A(I)[I];
            B(J)[I:M]:= B(J)[I:M] + T*A(I)[I:M];
        ENDF;
    ENDI;
    ENDF;
    FOR I:=1 TO IP DO
        X(I)[1:N] :=B(I)[1:N];
    ENDF;
IF JOB > 1 THEN
```

```
/*
 * COMPUTE THE RESIDUES
 */
/*
```

```
FOR I :=1 TO IP DO
    RSD(I)[1:H]:=0.0;
ENDF;
FOR I :=1 TO IP DO
    K:=N+1;
    RSD(I)[K:M]:=B(I)[K:M];
ENDF;
FOR K := N DOWNTO 1 DO
    FOR L :=1 TO IP DO
        T:=ACK)[K:M].DOT. RSD(L)[K:M];
        IF WK[K]=0 THEN
            IERR:=K; GO TO LAB4
        ENDI;
        T:=-T/ACK)[K];
        RSD(L)[K:M]:=RSD(L)[K:M] + T*ACK)[K:M];
        SUM[L]:=RSD(L)[1:M].DOT. RSD(L)[1:M];
    ENDF;
ENDF;
IF WT[1] > 0 THEN
    FOR I := 1 TO IP DO
        RSD(I)[1:M]:= RSD(I)[1:M]/WT[1:M];
    ENDF;
    WT[1:M]:=WT[1:M]*WT[1:M];
    ENDI;
ENDI;
```

IF JOB <>2 THEN

/*
/*
/*

COMPUTE THE SOLUTIONS

*/
*/
*/

12

FOR I:= N DOWNTO 2 DO
 IF WK[I]=0 THEN
 IERR:=I; GO TO LAB4
 ELSE
 K:=I-1;
 FOR J:= 1 TO IP DO
 X(J)[I]:= -X(J)[I]/WK[I];
 T:= -X(J)[I];
 X(J)[1:K]:= X(J)[1:K] + T*AC(I)[1:K];
 ENDF;
 ENDI;
ENDF;
FOR I:=1 TO IP DO
 IF WK[1]=0 THEN
 IERR:=1; GO TO LAB4
 ELSE
 X(I)[1]:= -X(I)[1]/WK[1];
 ENDI;
ENDF;

ENDI;

/*
/*
/*

SAVE THE TRANSFORMATION

*/
*/
*/

LAB4: FOR I:=1 TO N DO
 T:=AC(I)[1]; AC(I)[1]:= -WK[I]; WK[I]:=T;

ENDF;

ENDP;
ENDM;

~~END OF CODE~~

APPENDIX B**FORTRAN Coding of QR Algorithm**

SUBROUTINE QRASOS(MAXM,MAXN,M,N,IP,A,B,WT,JOB,X,RSD,SUM,WK,IERR) QRAS0010 14

C*****QRAS0020

C* *QRAS0030

C* PURPOSE *QRAS0040

C* TO SOLVE M SIMULTANEOUS EQUATIONS IN N UNKNOWNS WITH IP *QRAS0050

C* RIGHT HAND SIDES SO THAT THE SOLUTIONS ARE THE BEST POSSIBLE*QRAS0060

C* FIT IN THE LEAST SQUARES SENSE. THE ROUTINE USES HOUSE- *QRAS0070

C* HOLDER TRANSFORMATIONS TO PERFORM THE QR DECOMPOSITION *QRAS0080

C* OF THE COEFFICIENT MATRIX. *QRAS0090

C* *QRAS0100

C* USE *QRAS0110

C* *QRAS0120

C* CALL QRASOS(MAXM,MAXN,M,N,IP,A,B,WT,JOB,X,RSD,SUM,WK,IERR) *QRAS0130

C* *QRAS0140

C* PARAMETERS *QRAS0150

C* *QRAS0160

C* MAXM AN INPUT INTEGER SPECIFYING THE FIRST DIMENSION OF THE *QRAS0170

C* A,B, AND RSD ARRAYS IN THE CALLING PROGRAM. MAXM MUST *QRAS0180

C* BE GREATER THAN OR EQUAL TO M. *QRAS0190

C* *QRAS0200

C* MAXN AN INPUT INTEGER SPECIFYING THE FIRST DIMENSION OF THE *QRAS0210

C* X ARRAY IN THE CALLING PROGRAM. MAXN MUST BE GREATER *QRAS0220

C* THAN OR EQUAL TO N. *QRAS0230

C* *QRAS0240

C* M AN INPUT INTEGER SPECIFYING THE NUMBER OF ROWS OF THE *QRAS0250

C* A AND B ARRAYS. M MUST BE GREATER THAN OR EQUAL TO N. *QRAS0260

C* *QRAS0270

C* N AN INPUT INTEGER SPECIFYING THE NUMBER OF COLUMNS OF *QRAS0280

C* THE A ARRAY. *QRAS0290

C* *QRAS0300

C* IP AN INPUT INTEGER SPECIFYING THE NUMBER OF COLUMNS OF *QRAS0310

C* THE B ARRAY. *QRAS0320
 C* *QRAS0330
 C* A AN INPUT/OUTPUT TWO-DIMENSIONAL APRAY WITH FIRST DIMEN- *QRAS0340
 C* SION EQUAL TO MAXM AND SECOND DIMENSION AT LEAST N. *QRAS0350
 C* ON INPUT, A MUST CONTAIN THE MATRIX OF COEFFICIENTS OF *QRAS0360
 C* THE SYSTEM OF EQUATIONS. ON OUTPUT, A CONTAINS INFOR- *QRAS0370
 C* MATION DESCRIBING THE QR DECOMPOSITION OF A. *QRAS0380
 C* *QRAS0390
 C* B AN INPUT TWO-DIMENSIONAL ARRAY WITH FIRST DIMENSION *QRAS0400
 C* EQUAL TO MAXM AND SECOND DIMENSION AT LEAST IP. *QRAS0410
 C* THE COLUMNS OF B MUST CONTAIN THE IP RIGHT HAND SIDE *QRAS0420
 C* VECTORS. *QRAS0430
 C* *QRAS0440
 C* WT AN INPUT ONE-DIMENSIONAL ARRAY OF WEIGHTS. IF WEIGHT- *QRAS0450
 C* ING IS DESIRED, WT MUST HAVE LENGTH AT LEAST M, AND *QRAS0460
 C* THE FIRST M LOCATIONS MUST CONTAIN REAL NUMBERS GREATER *QRAS0470
 C* THAN ZERO. IF WEIGHTING IS NOT DESIRED, WT CAN CONSIST *QRAS0480
 C* OF A SINGLE LOCATION WHICH MUST CONTAIN A NEGATIVE REAL *QRAS0490
 C* NUMBER. *QRAS0500
 C* *QRAS0510
 C* JOB AN INPUT INTEGER SPECIFYING RESULTS TO BE COMPUTED. *QRAS0520
 C* *QRAS0530
 C* -1 COMPUTE SOLUTIONS ONLY. *QRAS0540
 C* -2 COMPUTE RESIDUALS ONLY. *QRAS0550
 C* -3 COMPUTE BOTH SOLUTIONS AND RESIDUALS. *QRAS0560
 C* *QRAS0570
 C* X AN OUTPUT TWO-DIMENSIONAL ARRAY CONTAINING THE SOLU- *QRAS0580
 C* TIONS. IF JOB=1 OR JOB=3, X MUST BE DIMENSIONED WITH *QRAS0590
 C* FIRST DIMENSION EQUAL TO MAXN AND SECOND DIMENSION *QRAS0600
 C* AT LEAST IP. IF JOB=2, X CAN BE A DUMMY PARAMETER. *QRAS0610
 C* *QRAS0620

C* RSD AN OUTPUT TWO-DIMENSIONAL ARRAY CONTAINING THE RESID-*QRAS0630
 C* UALS. IF JOB=2 OR JOB=3, RSD MUST BE DIMENSIONED WITH *QRAS0640 16
 C* FIRST DIMENSION EQUAL TO MAXM AND SECOND DIMENSION *QRAS0650
 C* AT LEAST IP. IF JOB=1, RSD CAN BE A DUMMY PARAMETER. *QRAS0660
 C* *QRAS0670
 C* SUM AN OUTPUT ONE-DIMENSIONAL ARRAY CONTAINING THE WEIGHTED *QRAS0680
 C* SUMS OF SQUARES OF THE RESIDUALS. IF JOB=2 OR JOB=3, *QRAS0690
 C* SUM MUST BE DIMENSIONED AT LEAST IP. IF JOB=1, SUM *QRAS0700
 C* CAN BE A DUMMY PARAMETER. *QRAS0710
 C* *QRAS0720
 C* WK A ONE-DIMENSIONAL WORK ARRAY WHICH MUST BE DIMENSIONED *QRAS0730
 C* AT LEAST N. ON OUTPUT, WK CONTAINS INFORMATION ON THE *QRAS0740
 C* QR DECOMPOSITION OF A. *QRAS0750
 C* *QRAS0760
 C* IERR AN INTEGER ERROR CODE. *QRAS0770
 C* *QRAS0780
 C* -0 NO ERROR DETECTED. *QRAS0790
 C* -1 N IS GREATER THAN M. *QRAS0800
 C* -2 THE DECOMPOSED MATRIX IS SINGULAR. *QRAS0810
 C* -3 WEIGHTING WAS REQUESTED AND ONE OR MORE WEIGHTS *QRAS0820
 C* IS NEGATIVE. *QRAS0830
 C* *QRAS0840
 C* REQUIRED ROUTINES NORMS,SQRDC2,SQRL2,SAXPY1,SDOT1,SSCAL *QRAS0850
 C* SCOPY *QRAS0860
 C* *QRAS0870
 C* FORTRAN FUNCTIONS ABS,AMAX1,MIN0,MOD,SIGN,SQRT *QRAS0880
 C* *QRAS0890
 C* SOURCE COMPUTER SCIENCES CORPORATION, *QRAS0900
 C* HAMPTON, VA. *QRAS0910
 C* *QRAS0920
 C* LANGUAGE FORTRAN *QRAS0930
 C* *QRAS0940

C*	DATE RELEASED	AUGUST 1, 1978	*QRAS0950 17
C*			*QRAS0960
C*	LATEST REVISION	OCTOBER 10, 1978	*QRAS0970
C*			*QRAS0980
C*			*QRAS0990
*****QRAS1000			
	DIMENSION A(MAXM,1),B(MAXM,1),X(MAXN,1),RSD(MAXM,1),WT(1),WK(1)		QRAS1010
	DIMENSION SUM(1)		QRAS1020
	IERR = 0		QRAS1030
C			QRAS1040
C			QRAS1050
C	CHECK FOR M LESS THAN N.		QRAS1060
C			QRAS1070
	IF(M .GE. N) GO TO 10		QRAS1080
	IERR = 1		QRAS1090
	GO TO 160		QRAS1100
C			QRAS1110
C	CHECK FOR NO WEIGHTING		QRAS1120
C			QRAS1130
	10 IF(WT(1) .LT. 0.0) GO TO 80		QRAS1140
C			QRAS1150
C	CHECK FOR ILLEGAL WEIGHTS		QRAS1160
C			QRAS1170
	DO 20 I = 2, M		QRAS1180
	IF(WT(I) .LE. 0.0) GO TO 30		QRAS1190
20	CONTINUE		QRAS1200
	GO TO 40		QRAS1210
30	IERR = 3		QRAS1220
	GO TO 160		QRAS1230
C			QRAS1240
C	WEIGHT THE A AND B ARRAYS BY THE SQUARE ROOT		QRAS1250

C OF THE WEIGHT ARRAY. QRAS1260
 C QRAS1270 18
 40 DO 70 I = 1, M QRAS1280
 WT(I) = SQRT(WT(I)) QRAS1290
 DO 50 J = 1, N QRAS1300
 AC(I,J) = WT(I)*AC(I,J) QRAS1310
 50 CONTINUE QRAS1320
 DO 60 J = 1, IP QRAS1330
 BC(I,J) = WT(I)*BC(I,J) QRAS1340
 60 CONTINUE QRAS1350
 70 CONTINUE QRAS1360
 80 CONTINUE QRAS1370
 C QRAS1380
 C CALL SQRDC2 TO DECOMPOSE A QRAS1390
 C QRAS1400
 CALL SORDC2(A,MAXM,M,N,WK) QRAS1410
 C QRAS1420
 C CALL SQRSL2 TO SOLVE FOR IP RIGHT HAND SIDES QRAS1430
 C QRAS1440
 CALL SQRSL2(A,MAXM,M,N,MAXN,IP,WK,B,X,RSD,JOB,IERR) QRAS1450
 IF(IERR .EQ. 0) GO TO 90 QRAS1460
 IERR = 2 QRAS1470
 GO TO 160 QRAS1480
 90 CONTINUE QRAS1490
 C QRAS1500
 C COMPUTE THE SUM OF WEIGHTED SQUARES OF RESIDUALS. QRAS1510
 C QRAS1520
 IF(JOB .EQ. 1) GO TO 140 QRAS1530
 DO 110 J = 1, IP QRAS1540
 SUM(J) = 0.0 QRAS1550
 DO 100 I = 1, M QRAS1560
 SUM(J) = SUM(J) + RSD(I,J)*RSD(I,J) QRAS1570

100	CONTINUE	QRAS1580
110	CONTINUE	QRAS1590
C		QRAS1600
C	COMPUTE UNWEIGHTED RESIDUALS	QRAS1610
C		QRAS1620
	IF(WT(1) .LT. 0.0) GO TO 160	QRAS1630
	DO 130 I = 1, M	QRAS1640
	DO 120 J = 1, IP	QRAS1650
	RSD(I,J) = RSD(I,J)/WT(I)	QRAS1660
120	CONTINUE	QRAS1670
130	CONTINUE	QRAS1680
140	CONTINUE	QRAS1690
	IF(WT(1) .LT. 0.0) GO TO 160	QRAS1700
	DO 150 I=1,M	QRAS1710
	WT(I) = WT(I)*WT(I)	QRAS1720
150	CONTINUE	QRAS1730
160	CONTINUE	QRAS1740
	RETURN	QRAS1750
	END	QRAS1760
	SUBROUTINE SQPDC2(X,LDX,N,P,QRAUX)	QRAS1770
	INTEGER LDX,N,P	QRAS1780
	REAL X(LDX,1),QRAUX(1)	QRAS1790
C		QRAS1800
C	SQPDC2 USES HOUSEHOLDER TRANSFORMATIONS TO COMPUTE THE QR	QRAS1810
C	FACTORIZATION OF AN N BY P MATRIX X.	QRAS1820
C		QRAS1830
C	ON ENTRY	QRAS1840
C		QRAS1850
C	X REAL(LDX,P), WHERE LDX .GE. N.	QRAS1860
C	X CONTAINS THE MATRIX WHOSE DECOMPOSITION IS TO BE	QRAS1870
C	COMPUTED.	QRAS1880

C QRAS1890

C LDX INTEGER. GRAS1900 20

C LDX IS THE LEADING DIMENSION OF THE ARRAY X. GRAS1910

C GRAS1920

C N INTEGER. GRAS1930

C N IS THE NUMBER OF ROWS OF THE MATRIX X. GRAS1940

C GRAS1950

C P INTEGER. GRAS1960

C P IS THE NUMBER OF COLUMNS OF THE MATRIX X. GRAS1970

C GRAS1980

C GRAS1990

C ON RETURN GRAS2000

C GRAS2010

C X X CONTAINS IN ITS UPPER TRIANGLE THE UPPER GRAS2020

C TRIANGULAR MATRIX R OF THE QR FACTORIZATION. GRAS2030

C BELOW ITS DIAGONAL X CONTAINS INFORMATION FROM GRAS2040

C WHICH THE ORTHOGONAL PART OF THE DECOMPOSITION GRAS2050

C CAN BE RECOVERED. GRAS2060

C GRAS2070

C QRAUX REAL(P). GRAS2080

C QRAUX CONTAINS FURTHER INFORMATION REQUIRED TO RECOVER GRAS2090

C THE ORTHOGONAL PART OF THE DECOMPOSITION. GRAS2100

C GRAS2110

C GRAS2120

C LINPACK SUBROUTINE SQRDC VEPSON DATED 07/14/77, REVISED BY GRAS2130

C COMPUTER SCIENCES CORPORATION, HAMPTON, VA. 10/10/78. GRAS2140

C GRAS2150

C BLAS SAXPY1,SDOT1,SSCAL LRC NORMS GRAS2160

C FORTRAN ABS,SIGN,SQRT,MOD GRAS2170

C GRAS2180

C INTERNAL VARIABLES GRAS2190

C GRAS2200

INTEGER J,L,LP1	QRAS2210
REAL SDOT,NRMXL,T	21 QRAS2220
C	QRAS2230
C	QRAS2240
C	QRAS2250
C PERFORM THE HOUSEHOLDEP REDUCTION OF X.	QRAS2260
C	QRAS2270
DO 190 L = 1, P	QRAS2280
GRAUX(L) = 0.0E0	QRAS2290
IF (L .EQ. N) GO TO 170	QRAS2300
C	QRAS2310
C COMPUTE THE HOUSEHOLDER TRANSFORMATION FOR COLUMN L.	QRAS2320
C	QRAS2330
NLEN = N-L+1	QRAS2340
CALL NORMS(NLEN,NLEN,1,X(L,L),2,NRMXL)	QRAS2350
IF (NRMXL .EQ. 0.0E0) GO TO 160	QRAS2360
IF (X(L,L) .NE. 0.0E0) NRMXL = SIGN(NRMXL,X(L,L))	QRAS2370
CALL SSCAL(N-L+1,1.0E0/NRMXL,X(L,L),1)	QRAS2380
X(L,L) = 1.0E0 + X(L,L)	QRAS2390
C	QRAS2400
C APPLY THE TRANSFORMATION TO THE REMAINING COLUMNS.	QRAS2410
C	QRAS2420
LP1 = L + 1	QRAS2430
IF (P .LT. LP1) GO TO 150	QRAS2440
DO 140 J = LP1, P	QRAS2450
T = -SDOT1(N-L+1,X(L,L),X(L,J))/X(L,L)	QRAS2460
CALL SAXPY1(N-L+1,T,X(L,L),X(L,J))	QRAS2470
140 CONTINUE	QRAS2480
150 CONTINUE	QRAS2490
C	QRAS2500
C	QRAS2510

C SAVE THE TRANSFORMATION. QRAS2520

C GRAUX(L) = X(L,L) QRAS2530 22

C X(L,L) = -NRMXL QRAS2540

160 CONTINUE QRAS2550

170 CONTINUE QRAS2560

180 CONTINUE QRAS2570

RETURN QRAS2580

END QRAS2590

SUBROUTINE SQRSL2(X,LDX,N,K,LDB,IP,QRAUX,Y,BETA,RSD,JOB,INFO) QRAS2600

INTEGER LDX,N,K,LDB,IP,JOB,INFO QRAS2610

REAL X(LDX,1),QRAUX(1),Y(LDX,1),BETA(LDB,1),RSD(LDX,1) QRAS2620

C QRAS2630

C SQRSL2 APPLIES THE OUTPUT OF THE SUBROUTINE SQRDC2 TO QRAS2640

C COMPUTE A SET OF IP LEAST SQUARES SOLUTIONS AND RESIDUALS. THE QRAS2650

C OUTPUT OF SQRDC2 IS THE DECOMPOSITION OF THE N BY K MATRIX QRAS2660

C X IN THE FORM QRAS2670

C QRAS2680

C X = Q * (R) QRAS2690

C (Q) QRAS2700

C QRAS2710

C WHERE Q IS ORTHOGONAL AND R IS UPPER TRIANGULAR. THIS QRAS2720

C INFORMATION IS CONTAINED IN CODED FORM IN THE ARRAY X QRAS2730

C AND THE ARRAY QRAUX. QRAS2740

C QRAS2750

C ON ENTRY QRAS2760

C QRAS2770

C X REAL(LDX,K), WHERE LDX .GE. N. QRAS2780

C X CONTAINS THE OUTPUT FROM SQRDC. QRAS2790

C QRAS2800

C LDX INTEGER. QRAS2810

C LDX IS THE LEADING DIMENSION OF THE ARRAY X. QRAS2820

C QRAS2830

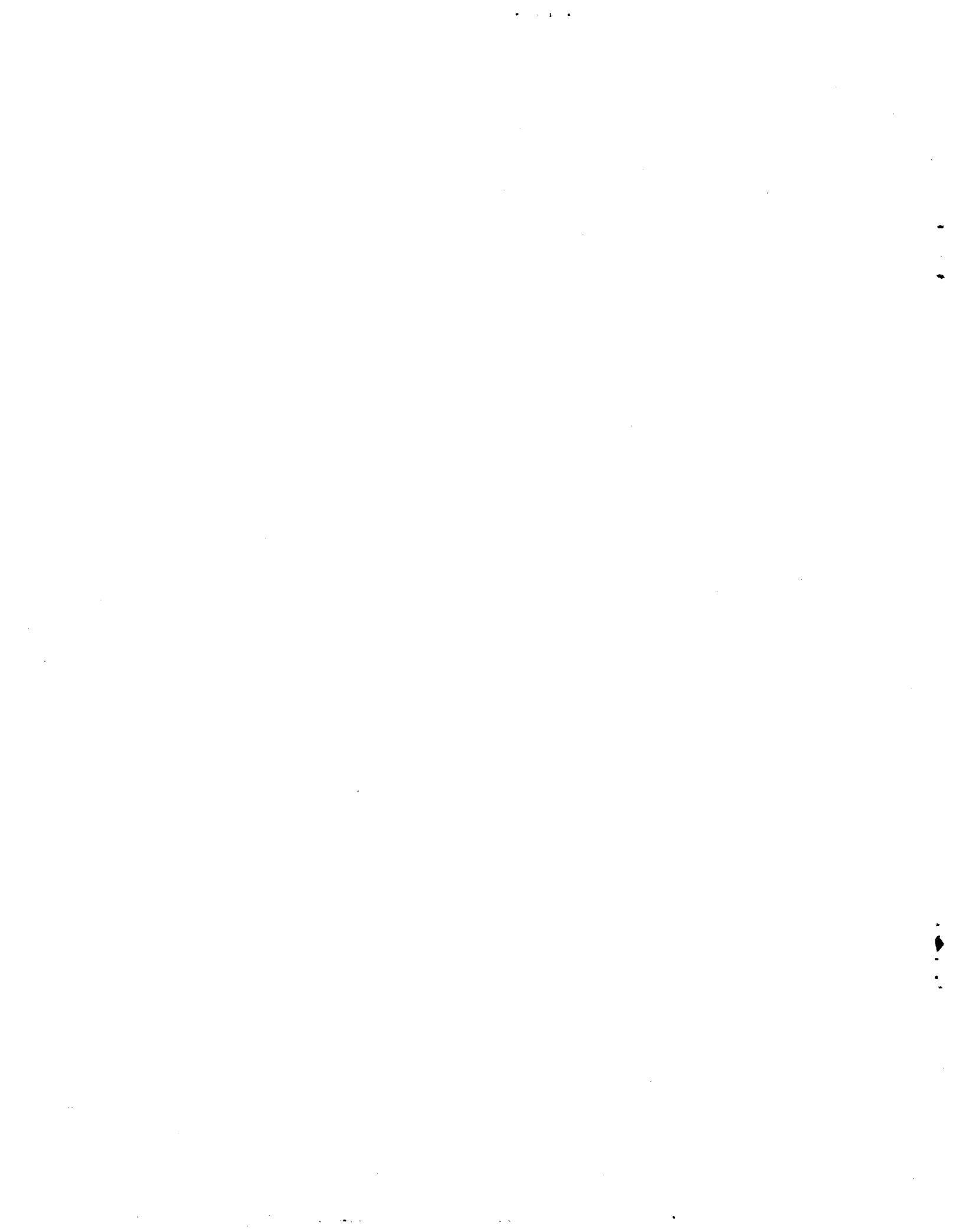
C	N	INTEGER.	QRAS2840
C		N IS THE NUMBER OF ROWS OF THE MATRIX X.	QPAS2850
C			QRAS2860
C	K	INTEGER.	QRAS2870
C		K IS THE NUMBER OF COLUMNS OF THE MATRIX X.	QRAS2880
C			QRAS2890
C	LDB	INTEGER.	QRAS2900
C		LDB IS THE LEADING DIMENSION OF THE ARRAY BETA.	QRAS2910
C			QRAS2920
C	IP	INTEGER.	QRAS2930
C		IP IS THE NUMBER OF RIGHT HAND SIDES.	QRAS2940
C			QRAS2950
C	QRAUX	REAL(K)	QRAS2960
C		QRAUX CONTAINS THE OUTPUT FROM SQRDC2.	QRAS2970
C			QRAS2980
C	Y	REAL(LDX,IP).	QRAS2990
C		Y IS THE N BY IP RIGHT HAND SIDE MATRIX THAT IS	QRAS3000
C		MANIPULATED BY SQRSL2.	QRAS3010
C			QRAS3020
C			QRAS3030
C	JOB	INTEGER.	QRAS3040
C		JOB IS A PARAMETER THAT CONTROLS WHAT IS TO BE	QRAS3050
C		COMPUTED.	QRAS3060
C			QRAS3070
C		IF JOB .EQ. 1 COMPUTE SOLUTIONS ONLY.	QRAS3080
C		IF JOB .EQ. 2 COMPUTE RESIDUALS ONLY.	QRAS3090
C		IF JOB .EQ. 3 COMPUTE SOLUTIONS AND RESIDUALS.	QRAS3100
C			QRAS3110
C		ON RETURN	QRAS3120
C			QRAS3130
C	BETA	REAL(LDB,IP).	QRAS3140

BETA CONTAINS THE SOLUTIONS OF THE LEAST SQUARES	QRAS3150
PROBLEMS	QRAS3160 24
MINIMIZE NORM2(Y(I) - X*BETAC(I)), I=1,2,...,IP	QRAS3170
IF THEIR COMPUTATION HAS BEEN REQUESTED.	QRAS3180
	QRAS3190
RSD REAL(LDX,1P)	QRAS3200
RSD CONTAINS THE LEAST SQUARES RESIDUALS	QRAS3210
Y(I) - X*BETAC(I), I=1,2,...,IP	QRAS3220
IF THEIR COMPUTATION HAS BEEN REQUESTED.	QRAS3230
	QRAS3240
INFO INTEGER	QRAS3250
INFO IS ZERO UNLESS THE CALCULATION OF BETA HAS BEEN	QRAS3260
REQUESTED AND P IS SINGULAR, IN WHICH CASE INFO IS	QRAS3270
THE INDEX OF THE FIRST ZERO DIAGONAL ELEMENT OF R.	QRAS3280
IN THIS CASE BETA IS UNALTERED.	QRAS3290
	QRAS3300
LINPACK SUBROUTINE QRSL VERSION DATED 07/14/77, REVISED BY	QRAS3310
COMPUTER SCIENCES CORPORATION, HAMPTON, VA. 10/10/78.	QRAS3320
	QRAS3330
BLAS SAXPY1,SCOPY,SDOT1	QRAS3340
FORTRAN ABS,MIN0,MOD	QRAS3350
	QRAS3360
INTERNAL VARIABLES	QRAS3370
	QRAS3380
INTEGER I,J,JJ,JU,KP1	QRAS3390
REAL SDOT,T,TEMP	QRAS3400
	QRAS3410
	QRAS3420
SET INFO FLAG	QRAS3430
	QRAS3440
INFO = 0	QRAS3450
JU = MIN(CK,N-1)	QRAS3460

C QRAS3470
 C 25
 C SPECIAL ACTION WHEN N=1 QRAS3480
 C QRAS3490
 IF (JU .NE. 0) GO TO 20 QRAS3500
 IF(XC1,1) .NE. 0.0) GO TO 5 QRAS3510
 INFO = 1 QRAS3520
 GO TO 220 QRAS3530
 5 CONTINUE QRAS3540
 DO 10 L = 1, IP QRAS3550
 IF(JOB .NE. 2) BETAC(1,L) = Y(1,L)/XC1,1) QRAS3560
 IF(JOB .NE. 1) RSD(1,L) = 0.0E0 QRAS3570
 10 CONTINUE QRAS3580
 GO TO 220 QRAS3590
 20 CONTINUE QRAS3600
 C QRAS3610
 C COMPUTE TRANS(Q)*Y QRAS3620
 C QRAS3630
 DO 50 J = 1, JU QRAS3640
 IF (QRAUX(J) .EQ. 0.0E0) GO TO 40 QRAS3650
 TEMP = XCJ,J)
 XCJ,J) = QRAUX(J) QRAS3660
 QRAS3670
 DO 30 L = 1, IP QRAS3680
 T = -SDOT1(N-J+1,XCJ,J),YCJ,L))/XCJ,J) QRAS3690
 CALL SAXPY1(N-J+1,T,XCJ,J),YCJ,L)) QRAS3700
 30 CONTINUE QRAS3710
 XCJ,J) = TEMP QRAS3720
 40 CONTINUE QRAS3730
 50 CONTINUE QRAS3740
 KP1 = K + 1 QRAS3750
 IF (JOB .EQ. 1 .OR. K .EQ. N) GO TO 70 QRAS3760
 DO 60 L = 1, IP QRAS3770

	CALL SCOPY(K,Y(KP1,L),1,RSD(KP1,L),1)	QRAS3780
60	CONTINUE	QRAS3790 26
70	CONTINUE	QRAS3800
	IF (JOB .EQ. 2) GO TO 120	QRAS3810
C		QRAS3820
C	COMPUTE BETA	QRAS3830
C		QRAS3840
	DO 75 L = 1, IP	QRAS3850
	CALL SCOPY(K,Y(1,L),1,BETAC1,L),1)	QRAS3860
75	CONTINUE	QRAS3870
	DO 100 JJ = 1, K	QRAS3880
	J = K - JJ + 1	QRAS3890
	IF (X(J,J) .NE. 0.0E0) GO TO 80	QRAS3900
	INFO = J	QRAS3910
CEXIT	QRAS3920
	GO TO 220	QRAS3930
80	CONTINUE	QRAS3940
	DO 95 L = 1, IP	QRAS3950
	BETAC(J,L) = BETAC(J,L)/X(J,J)	QRAS3960
	IF (J .EQ. 1) GO TO 90	QRAS3970
	T = -BETAC(J,L)	QRAS3980
	CALL SAXPY1(J-1,T,X(1,J),BETAC1,L))	QRAS3990
90	CONTINUE	QRAS4000
95	CONTINUE	QRAS4010
100	CONTINUE	QRAS4020
110	CONTINUE	QRAS4030
120	CONTINUE	QRAS4040
	IF (JOB .EQ. 1) GO TO 210	QRAS4050
C		QRAS4060
C	COMPUTE RSD IF REQUIRED	QRAS4070
C		QRAS4080
	DO 160 L = 1, IP	QRAS4090

DO 150 I = 1, K	GRAS4100
RSD(I,L) = 0.0E0	GRAS4110
150 CONTINUE	GRAS4120
160 CONTINUE	GRAS4130
DO 200 JJ = 1, JU	GRAS4140
J = JU - JJ + 1	GRAS4150
IF (GRAUX(JJ) .EQ. 0.0E0) GO TO 190	GRAS4160
TEMP = XCJ,JJ	GRAS4170
XCJ,JJ = GRAUX(JJ)	GRAS4180
DO 170 L = 1, IP	GRAS4190
T = -SDOT1(H-J+1,XCJ,JJ,RSD(J,L))/XCJ,JJ	GRAS4200
CALL SAXPY1(H-J+1,T,XCJ,JJ,RSD(J,L))	GRAS4210
170 CONTINUE	GRAS4220
XCJ,JJ = TEMP	GRAS4230
190 CONTINUE	GRAS4240
200 CONTINUE	GRAS4250
210 CONTINUE	GRAS4260
220 CONTINUE	GRAS4270
RETURN	GRAS4280
END	GRAS4290



1. Report No. NASA CR-165730	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle STAR-ADAPTATION OF QR ALGORITHM		5. Report Date June 1981	
7. Author(s) SHANTILAL N. SHAH		6. Performing Organization Code	
9. Performing Organization Name and Address Hampton Institute Hampton, Virginia 23668		8. Performing Organization Report No.	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546		10. Work Unit No.	
		11. Contract or Grant No.	
		13. Type of Report and Period Covered Contractor Report	
		14. Sponsoring Agency Code	
15. Supplementary Notes This work was performed under Cooperative Agreement NCC1-7 with the NASA Langley Research Center and Hampton Institute.			
16. Abstract The QR algorithm used on a serial computer and presently executed on the Control Data Corporation 6000 Computer was adapted to execute efficiently on the Control Data STAR-100 computer. This paper describes how the scalar program was adapted for the STAR-100 and indicates why these adaptations yielded an efficient STAR program. Program listings of the old scalar version and the new vectorized SL/l version are presented in the appendices. Execution times for the two versions applied to the same system of linear equations, are compared.			
17. Key Words (Suggested by Author(s)) STAR computer QR algorithm SL/l, LINPACK, BLAS		18. Distribution Statement Unclassified - Unlimited Subject Category 64	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 28	22. Price A03

